An Engineering Discipline of Software: an Australian perspective

Jocelyn Armarego & Robert Cohen
Curtin University of Technology, Bentley, Western Australia 6107
University of Newcastle, Callaghan, New South Wales 2308
Email: jocelyn@cs.curtin.edu.au, rfc@cs.newcastle.edu.au

Abstract
Engineering faculties within Australian universities are being asked to explore the feasibility of mounting separate Software Engineering programmes. This paper provides an overview of the background for the development of such programmes, the accreditation requirements imposed and the difference in perception of Software Engineering between the Australian and the United States engineering professions. A survey of the curricula of Bachelor of Engineering (Software Engineering) programmes in Australian universities was undertaken, and the programmes examined from the perspectives of requirements for Australian engineering accreditation and the models for undergraduate education incorporating Software Engineering.

1 Introduction

The 1990s is seeing the development, within Australian universities, of separate Software Engineering programmes. While some universities initiate major streams in Software Engineering within existing Computer Science degrees, the current trend is to offer separate programmes within Engineering faculties. A number of universities have explored the feasibility of mounting such programmes.

Philosophically, one of the principle reasons for advocating the development of separate Software Engineering programmes is the perception that Computer Science curricula have evolved to a state that does not adequately prepare students for professional careers building software-intensive systems. Amongst other reasons, this is seen to be due to a leaning towards science (and research) and away from engineering (and professional practice) in CS curricula, due to defacto adherence to ACM curriculum guidelines [1] (Ford [2]), and presumably its revised versions.

The point that is made by the Software Engineering Institute (SEI) [3] is that Software Engineering education will not be achieved by adding Software Engineering concepts and techniques to Computer Science programmes - an

engineering approach to the whole curriculum is necessary.

2 Accreditation for Software Engineering

As Software Engineering programmes begin to emerge, the pressure for accreditation follows: the goal of accreditation is to define the *minimum* standard for programmes, and is especially significant in disciplines leading to professional practice, such as engineering.

The Institution of Engineers Australia (IEAust) is the qualifying body for professional engineers in Australia, with programme accreditation based on, amongst other guidelines, their document, *Basic Requirements for a Professional Engineering Course* (IEAust [4]).

In 1985 an IEAust working party, established to consider Software Engineering as a professional discipline, reported its findings. This document (Dixon Hughes [5]) continues to define the current philosophy for the development of engineering programmes incorporating Software Engineering, in Australian universities. The findings of primary interest are

- the Working Party does not support the establishment of undergraduate Engineering degrees in Software Engineering but accepts Software Engineering as a core component of Computer Systems Engineering (CSE) programmes.
  The IEAust view is that Software Engineering of itself is a specialist activity within the computer field rather than an engineering discipline
- a preference for the term Computer Systems Engineering as more appropriate for the concept of Computer and Software Engineering.
  The following definition of Computer Systems Engineering (CSE) is provided

  > CSE is the professional engineering discipline which covers the activities required to create a computing system to achieve an end-purpose in its own right and includes the design, construction and effective integration of hardware and/or software components
  > (corrigenda to Dixon Hughes [5]).

  Note should be taken of the inclusion of a hardware component, and a comparison made to the SEI definition of Software Engineering:

  > - engineering is the systematic application of scientific knowledge in creating and building cost-effective solutions to practical problems in the service of mankind
  > - software engineering is that form of engineering that applies the principles of computer science and mathematics to achieving cost-effective solutions to software problems.

  This definition and its elaborations are discussed in Ford [3]

Discussions with IEAust, and in particular the Chairman of the Working Party on Software Engineering, who is currently Discipline Chair for Computer Systems and Information Technology for IEAust, indicate the present status:

- the report was ratified by the IEAust, but not all recommendations acted upon. In particular, Recommendation 6, which required IEAust to establish a

set of guidelines for the accreditation of CSE programmes in Australian Engineering schools, is only now (1996) to be investigated, and a working party established
- this working party is expected to revisit the recommendations of the 1985 report, but is not expected to modify, for example, the definition of CSE.

The IEAust view is that, from the perspective of an engineering discipline, CSE is seen to encompass those aspects of Software Engineering required by engineers, with the essential difference between SE and CSE being the ability to design and implement total systems which could involve hardware and/or software components. However, this approach to Software Engineering, as a specialist activity within the computer field rather than an engineering discipline, differs from the recent IEEE and ACM view. The current status of the US scene is that the ACM and IEEE Computer Society have joined forces to move Software Engineering towards professional status, perhaps by the end of the century (Jones [6]). This will allow it to be the 37th engineering profession, licenced and recognised.

The implication of the IEAust findings is that, at present, all Bachelor of Engineering (Software Engineering) degrees applying for IEAust accreditation do so on the basis of Dixon Hughes [5] and IEAust guidelines (IEAust [4]). The latter references the *IEEE Computer Society Model Programs in Computer Science and Engineering* (IEEE [7]) as the most useful starting point for Australian tertiary educational institutions seeking to introduce four-year programmes incorporating Software Engineering: it is seen to meet general accreditation standards similar to those in force in Australia (Dixon Hughes [5]). The point should be made, however, that the IEEE focus is Computer Science and Engineering programmes which *may* incorporate a component of Software Engineering, rather than SE education per se.

3 Model Curricula

The IEEE [7] model (summarised in Cain [8]) advocates a programme breakdown to achieve ABET (Accreditation Board for Engineering and Technology Inc, the sole US agency responsible for accreditation of engineering degrees) accreditation. This is shown as *Table 1*.

More recently, the ACM/IEEE-CS Joint Curriculum Task Force (ACM [9] , Turner [10]) placed computing simultaneously within the mathematical, scientific and engineering disciplines. The discipline of computing comprises nine subject areas, each of which contains fundamental subjects designated *common requirements* (see *Table 2).*

Of interest to this discussion is the allocation of 16% to Software Methodology & Engineering, though other components viewed as integral to successful undergraduate programmes could be considered Software Engineering issues. These include *abstraction* and *design*, (noted by Denning [11] as two of the three paradigms which characterise the discipline of computing), which permeate all nine subject areas, and a number of *recurring concepts* (defined as significant ideas, concerns processes and principles that unify an academic disci-

pline (Turner [10])), such as conceptual and formal models, levels of abstraction, reuse, to name a few.

Subject Area	Credit_hours	% of Programme
Science	16	11.9
Mathematics	18	13.4
Humanities and Social Science (including English composition)	18	13.4
required Computer Science and Engineering	31	23.1
Computer Science and Engineering electives	27	20.2
other Engineering	12	9
free electives	12	9
Total	134	100

Less than 5% of the programme could be categorised as Software Engineering. However, it is possible to focus on SE within the CS & E electives, increasing the Software Engineering component to a maximum of 25% of the programme.

Table 1: IEEE (1983) Sample Curriculum for ABET Accreditation

Subject Area	Lecture Hours (approx)	Percentage of common requirement
Algorithms & Data Structures	47	17.3
Architecture	59	22
AI & Robotics	9	3.3
Database & Information Retrieval	9	3.3
Human-Computer Communications	8	3
Numeric & Symbolic Computation	7	2.6
Operating Systems	31	11.4
Programming Languages	46	16.9
Software Methodology & Engineering	44	16.2
Social, Ethical & Professional Issues	11	4
Total	271	100

Table 2: ACM Common Requirements for Computing

Useful for our discussions are sample curricula for ABET accreditation based on this model:

Subject Area	Curriculum A		Curriculum B	
	Credit hours	% of Programme	Credit hours	% of Programme
Science	15	11	12	9
Mathematics	22	16	21	15
Humanities & Social Science (including English Composition)	24	18	24	18
Required Computer Science & Engineering	38	28	40	29
Computer Science & Engineering Electives (a sample specialisation in SE is demonstrated)	18	13	18	13
Other Engineering	9	7	9	7
Free Electives	9	7	12	9
Total	135	100	136	100

The Software Engineering component of these programmes varies from the common requirement of 16.2% of the required CS & E courses (ie 4.5% of programme) up to 17.5% of programme where the specialisation is also Software Engineering

Table 3: ACM/IEEE-CS (1991) Sample Curricula for ABET Accreditation

Within the same time frame, the SEI advocated the development of undergraduate Software Engineering programmes (Ford [3]). A curriculum model that reflects the spirit of both ABET and CSAB (Computing Sciences Accreditation Board) guidelines is proposed:

Subject Area	Credit_hours	% of Programme
Science	9	7.5
Mathematics	18	15
Humanities & Social Science (including English Composition)	30	25
Required Software Engineering (including Computer Science)	42	35
Software Engineering Electives	3	2.5
Free Electives	18	15
Total	120	100

Note: assume that the percentages extrapolate to a 135 credit-hour programme.
The CS component is 21.4% of the required Software Engineering courses (7.5% of whole programme)

Table 4: SEI (1990) Model Curriculum for ABET Accreditation

The IEAust does not prescribe the content of Bachelor of Engineering programmes in Australian universities. The guidelines are defined in terms of categories, some of which are assigned preferred percentages:

Category	Subject Area
1	mathematics, physical and other basic sciences, and computing
2	engineering science material, not entirely in one field
3	engineering synthesis or design and related communications skills
4	engineering applications material, including project work
5	basic principles underlying management of physical, human & financial resources
6	professional responsibility, social effects & ethical aspects of engineering practice
7	not less than 12 weeks of practical experience relevant to engineering

IEAust suggest that categories 1 and 2 dominate the curriculum, and categories 5 and 6 are valued at 10% and 15% of the total

Table 5: IEAust Curriculum Guidelines

However, accreditation is partly based on programme content.

Category	Subject Area
1a	mathematics, physical and other basic sciences
1b	computing, including computing science, excluding software engineering
2	engineering science material, specifically not design or lab work
3	engineering synthesis or design and related communications skills
4	engineering applications material, specifically project work and laboratory
5	basic principles underlying management of physical, human & financial resources
6	professional responsibility, social effects & ethical aspects of engineering practice and other free electives

Table 6: Breakdown of IEAust Curriculum Guidelines

The IEAust categories are further broken down as shown in *Table 6* in order to allow a comparison between IEAust guidelines and the model curricula, and allowing for the need to allocate courses to these categories primarily

based on syllabus information.

4 Bachelor of Engineering (SE) programmes at Australian universities

Within Australia there are now a handful of Engineering degrees which focus on Software Engineering, with other universities (including Curtin University of Technology) in the process of developing such programmes.

From the point of view of Software Engineering education, there are pragmatic advantages in mounting BE(SE) degrees - undergraduate Computer Science programmes are in general offered as 3-year Bachelor of Science degrees, the extra year of study to allow for the the range of science courses, and the completion of laboratory components, being a feature of Engineering programmes. In addition, funding of Engineering Departments is reported at a higher level that for Science Departments (Hudson [12]). Of these programmes, to date only one has been accorded full recognition by IEAust - accreditation being dependant on the stage of programme development, and requiring approximately five years.

University	Accreditation Level
University of Melbourne (MU)	Full
Murdoch University (MdU)	Preliminary
Swinburne University of Technology (SUT)	not yet submitted
University of Newcastle (UN)	not yet accredited
University of New South Wales (UNSW)	under development
Griffiths University (GU)	not yet submitted
Curtin University of Technology (CUT)	under development

Table 7: BE(SE) programmes at Australian Universities (Nov 1996)

The breakdown of the syllabi for these programmes to the categories required by the model curricula produces some interesting comparisons:

	Percentage of Programme					
IEAust	MU	MdU	SUT	UN	GU	CUT
1a	25	19.8/27.6	7.5	18.8	12.5	15/17.5
1b	23	18.2	20	31.3	18.8	32.5/30
2	8.9	12.5/5.2	17.5/22.5	9.4	15.6	15
3	8.9	23.4	35	18.8	21.9	22.5
4	13.5	13	10	12.5	12.5	7.5
5	16.4	7.4	5	6.3	12.5	2.5
6	4.6	5.2	5/0	3.1	6.3	5

All programmes include a professional practice component

Table 8: BE(SE) Curricula in IEAust categories

The IEAust guidelines place computing within Category 1, with mathematics and the sciences. However, Computer Science is elsewhere (Ford [3]) considered an Engineering science, which would boost the Category 2 figures above. Software Engineering courses predominate Categories 3 and 4, but may be allocated to other categories as appropriate.

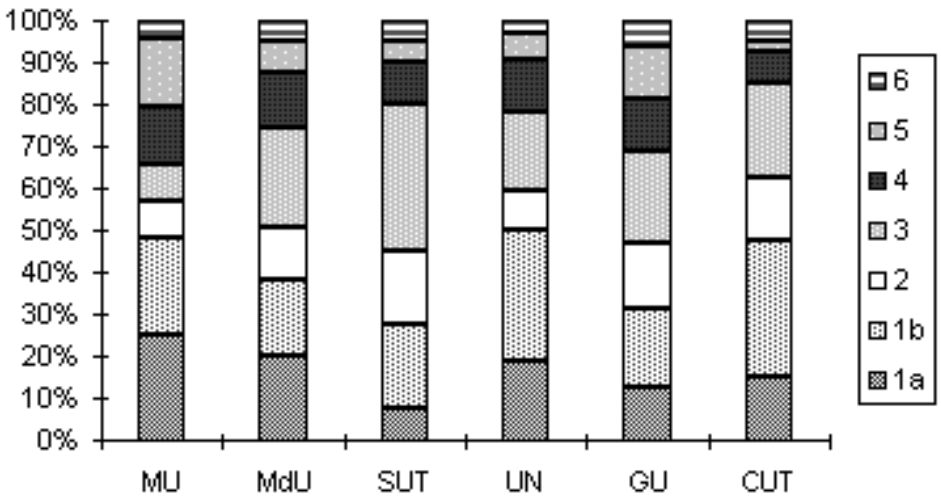

Figure 1: BE(SE) Curricula in IEAust categories

In terms of Subject Areas that could be aligned to the model curricula, programmes in Australian universities exhibit the following characteristics:

Subject Area	Percentage of Programme					
	MU	MdU**	SUT	UN	GU	CUT
Science	8.9	7.8	2.5	5.6	3.1	5
Maths	16.1	12	5	11.1	9.4	10/12.5
Humanities & SocSci	9	10	5/10	8.3	12.5	5
required CS & Eng	25	31.2	42.5	44.4	34.4	25/27.5
required SE	25.9	31.2	40	22.2	40.6	20
CS & Eng electives	10.5	7.8	5/0	8.3	-	30!
free electives	4.6	-	-*	-	-	2.5

* free electives are confined to specific engineering or social science areas, and are therefore included in those categories

** MdU organises courses based on modules, which may be taken from a number of subject areas. 12 credit points is equivalent to one full-time semester of study. In the table above, the percentage figures are from Roy [13]

! This is envisaged as a 'plug-in' component of the programme. The working curriculum uses CS communications and communications engineering as a sample.

Table 9: Australian BE(SE) programmes in model curricula categories

Of interest from the Australian perspective is that the IEAust differs markedly from the current models, both in the placing of Software Engineering within the profession, and in its association with other engineering disciplines. This affects programme design. Discussions with programme controllers suggest that the IEAust requirement for breadth impacts heavily on the content of BE(SE) programmes, and consequently leads to decisions on what Software Engineering material to leave out.

In addition, many of the programmes offered at Australian universities show their origins - an engineering viewpoint imposed on what is basically a Computer Science curriculum. Very few could be said to adhere more than in

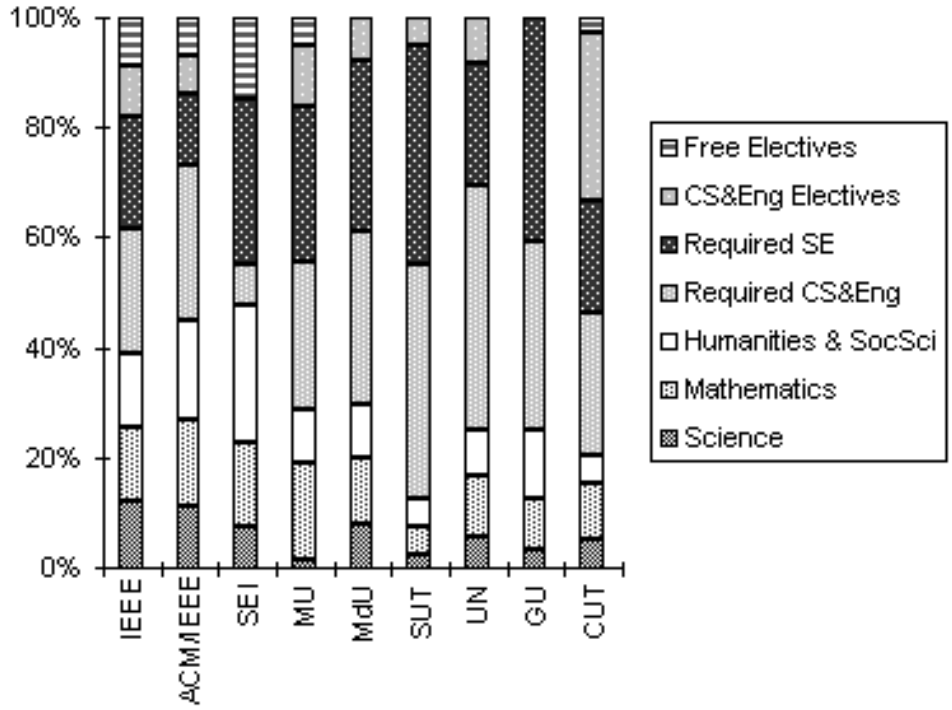

Figure 2: Australian BE(SE) programmes compared to the model curricula

the most general terms to the SEI model - the breadth and depth of SE courses advocated by that model may not yet be achievable given the youth of the discipline within the country.

5 Conclusion

While in other circles there is some suggestion that a model Software Engineering education may be achievable in no less than seven years of study (Davis [14]), universities in Australia are moving towards the acceptance that it requires more than the three years offered by Computer Science programmes. Within the current pragmatic climate, this is being achieved by seeking IEAust accreditation for BE(SE) programmes. However, given that SE education may not be achieved by adding SE concepts to Computer Science programmes, the question should be asked - will SE education be achieved where the adherence to an engineering approach is too strict?

References

1. ACM, Curriculum Committee on Computer Science (1968) Curriculum 68: recommendations for the undergraduate program in Computer Science. *Communications of the ACM* **11**(3) pp. 151-197
2. Ford, Garry (1994) *A Progress Report on Undergraduate Software Engineering Education*. Pittsburg (Penn): Software Engineering Institute Technical Report CMU/SEI-94-TR-11

3. Ford, Garry (1990) *1990 SEI Report on Undergraduate Software Engineering Education*. Pittsburg (Penn): Software Engineering Institute Technical Report CMU/SEI-90-204
4. IEAust (1993) *Basic Requirements for a Professional Engineering Course*. Canberra: Institution of Engineers, Australia Council
5. Dixon-Hughes, J R (1985) *Perspectives on Engineering and Computing Systems: report to the Institution of Engineers, Australia, by the Working Party on Software Engineering*. Canberra: Institution of Engineers, Australia
6. Jones, Caper (1995) Legal status of Software Engineering. *IEEE Computer* **28**(5) pp. 98-99
7. IEEE, Computer Society (1983) *Model Programs in Computer Science and Engineering*. Los Alamitos (Ca): Computer Society Press
8. Cain, J T, Langdon, G G, and Varanasi, M R (1984) The IEEE Computer Society model program in computer science and engineering. *IEEE Computer* **17**(4) pp. 8-17
9. ACM (1991) *Computing Curriculum 1991: report of the ACM/IEEE-CS Joint Curriculum Task Force*. New York: ACM
10. Turner, A Joe (1991) A summary of the ACM/IEEE-CS Joint Curriculum Task Force Report: Computing Curricula 1991. *Communications of the ACM* **34**(6) pp. 69-84
11. Denning, Peter J and et al (1989) Computing as a discipline. *Communications of the ACM* **32**(1) pp. 9-23
12. Hudson, Hugh (chair) (1992) *Report of the Discipline Review of Computing Studies and Information Sciences Education*. Canberra: Australian Government Publishing Service
13. Roy, Geoffrey G and Veraart, Valerie E (1996) Software Engineering Education: from an engineering perspective. *In Software Engineering: Education & Practice. Proceedings* **1996** pp. 256-262 Dunedin Computer Society Press
14. Davis, Alan (1996) From the Editor: practitioner, heal thyself. *IEEE Software* **13**(3) p. 4